REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
14–08–2002	Technical	

4. TITLE AND SUBTITLE

AN ARTIFICIAL NEURAL NETWORK TRACKING ARCHITECTURE

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0601152N

6. AUTHORS

M. W. Owen
SSC San Diego

A. R. Stubberud
UC Irvine

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

SSC San Diego
San Diego, CA 92152-5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217–5000

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this paper, a neural extended Kalman filter algorithm is used for tracking of a highly maneuvering target. The neural extended Kalman filter is used to improve a mathematical motion model for use in prediction. Instead of just applying a high process noise model (a catch-all technique) in an interacting multiple model architecture to hold a target through a maneuver, a neural extended Kalman filter is used to predict the correct velocity and acceleration states of a target. This, in turn, may allow noise reduction during a target maneuver. Tracking results that stress the algorithm during severe maneuvers are shown along with the tuning parameter issues of the artificial neural network.

Presented at IRIS National Symposium on Sensor and Data Fusion (NSSDF) 2002 Military Sensing Symposia (MSS).

15. SUBJECT TERMS

Mission Area: Surveillance

tracking systems
Mathematical motion model
neural-extended Kalman filter
multiple model architecture
noise reduction

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			M. W. Owen
U	U	U	UU	20	19B. TELEPHONE NUMBER *(Include area code)* (619) 553-2041

An Artificial Neural Network Tracking Architecture

August 14, 2002

Mark W. Owen
SPAWAR Systems Center San Diego
Code 2725
53560 Hull Street
San Diego, CA 92125
mowen@spawar.navy.mil

Allen R. Stubberud
Department of Electrical and Computer Engineering
University of California, Irvine
Irvine, CA 92697
arstubbe@uci.edu

ABSTRACT

In previous work, the authors embedded a neural extended Kalman filter algorithm in a closed loop feedback system and presented results on a servo-tracking problem. The reason behind using a neural extended Kalman filter algorithm was to improve the estimator's mathematical model. Without the correct model the unmodeled dynamics or nonlinearities in the system would yield an unstable filter.

In this paper, a neural extended Kalman filter algorithm is used for tracking of a highly maneuvering target. The neural extended Kalman filter is used to improve a mathematical motion model for use in prediction. Instead of just applying a high process noise model (a catch all technique) in an interacting multiple model architecture to hold a target through a maneuver, a neural extended Kalman filter is used to predict the correct velocity and acceleration states of a target. This in turn may allow noise reduction during a target maneuver.

Tracking results that stress the algorithm during severe maneuvers will be shown along with the tuning parameter issues of the artificial neural network.

1. Introduction

The Robust Tracking with a Neural Extended Kalman Filter (NEKF) project is an Office of Naval Research (ONR) In-House Laboratory Independent Research (ILIR) sponsored effort at SPAWAR Systems Center San Diego [1]. The project's goal is to provide an improved state estimation capability for current U.S. Navy tracking systems. The NEKF will provide added capability for the real-time modeling of maneuvers and, therefore, will enhance the ability of tracking systems to adapt appropriately.

The NEKF has been used in the past in control system technology and for system identification [2, 3, 4, 5, 6, 7, and 8]. During this project the NEKF will be incorporated into a tracking architecture to provide robust tracking capabilities that are currently unavailable.

2. Background

State estimation and tracking of highly maneuvering targets has and still is an extremely difficult task in modern tracking systems. Current state estimation approaches to the tracking problem include alpha-beta filters, Kalman filters, interacting multiple model (IMM) filters, probabilistic data association (PDA) trackers, joint PDA (JPDA) trackers to name a few [9 and 10]. State estimation is the problem of deriving a set of system states that are of interest to a system or decision maker. System states consist of parameters such as position, velocity, frequencies, magnetic moments, and other attributes of interest. Most often system states are not measurable at the system output. For example, range and bearing of a target may be available from a radar sensor but the position and velocity of the target needs to be derived from the radar measurement. To derive these states an estimation algorithm is used. A mathematical system model is necessary for the aforementioned filter algorithms to perform state estimation.

A well known state estimation algorithm is the tried and true Kalman filter developed four decades ago by R. E. Kalman [11]. The Kalman filter is widely used in government and industry tracking problems. The Kalman filter uses an assumed mathematical system model (i.e. a straight line motion model for an aircraft) to estimate the states (position, velocity, signatures, etc...) of the aircraft. A problem occurs when the aircraft or system being tracked changes from the assumed motion model. The filter will tend to lag behind the true state of the target or even diverge and become unstable and unable to estimate the system. A block diagram of the Kalman filter is shown in Figure 1 below. The filter consists of the dynamic system to be tracked, a mathematical system model $\mathbf{F}(k)$, an observation model $\mathbf{H}(k+1)$, the Kalman gain $\mathbf{K}(k+1)$, a predicted observation $\hat{\mathbf{z}}(k+1)$, and the system state vector $\hat{\mathbf{x}}(k \mid k)$. The Kalman filter is a recursive estimation algorithm that is driven by the observation data $\mathbf{z}(k+1)$ from the dynamic system. As data is processed a filtered estimate of the state $\hat{\mathbf{x}}(k \mid k)$ is generated by the filter. As long as the dynamic system is linear, and matches the assumed motion model, the filter will perform quite well and tracking will be successful. In cases where motion model and/or the observation model are nonlinear, then an extension of the linear Kalman filter must be used. An extension of the Kalman filter is the extended Kalman filter (EKF) [12]. The EKF is used to handle known nonlinearities. The extended

Kalman filter evaluates the nonlinear system model $\mathbf{f}\left(\mathbf{x}(k),k\right)$ about the current state and the nonlinear observation model $\mathbf{h}\left(\mathbf{x}(k+1),k+1\right)$ about the predicted state and $\mathbf{H}(k+1)$ is the Jacobian of the nonlinear observation equations evaluated at the predicted state. The extended Kalman filter equations are shown below in Figure 2. The Kalman filter is the same except for the nonlinear equations mentioned previously.

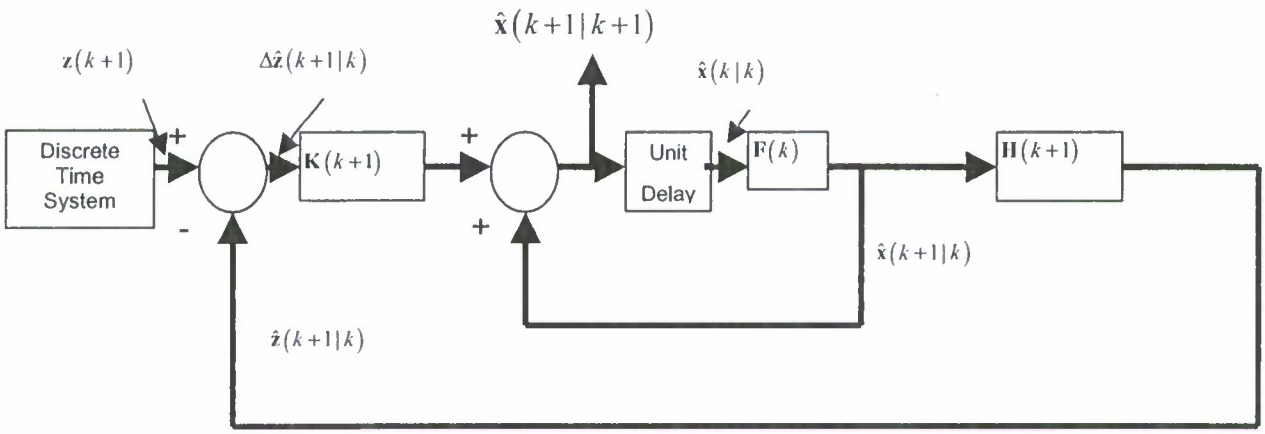

Figure 1. Kalman Filter Block Diagram

System Model

$$\mathbf{x}(k+1)=\mathbf{f}\left[\mathbf{x}(k),k\right]+\mathbf{L}\left[\mathbf{x}(k),k\right]\mathbf{w}(k)$$
$$\mathbf{z}(k+1)=\mathbf{h}\left[\mathbf{x}(k+1),k+1\right]+\mathbf{v}(k+1)$$

Initialization

$$\mathbf{x}(0|0)=E\left[\mathbf{x}(0)\right]$$
$$\mathbf{P}(0|0)=\mathbf{P}(0)$$

Prediction Equations

$$\hat{\mathbf{x}}(k+1|k)=\mathbf{f}\left[\hat{\mathbf{x}}(k|k),k\right]$$
$$\mathbf{P}(k+1|k)=\mathbf{A}(k)\mathbf{P}(k|k)\mathbf{A}'(k)+\mathbf{L}(k)\mathbf{Q}(k)\mathbf{L}'(k)$$

where

$$\mathbf{Q}(k)=E\left[\mathbf{w}(k)\mathbf{w}'(k)\right]$$
$$\mathbf{A}(k)=\frac{\partial\mathbf{f}\left[\mathbf{x}(k)\right]}{\partial\mathbf{x}(k)}\Big|_{\mathbf{x}(k)=\hat{\mathbf{x}}(k|k)}$$
$$\mathbf{L}(k)=\mathbf{L}\left[\hat{\mathbf{x}}(k|k),k\right]$$

Update Equations

$$\hat{\mathbf{x}}(k+1|k)=\mathbf{f}\left[\hat{\mathbf{x}}(k|k),k\right]$$
$$\mathbf{P}(k+1|k)=\mathbf{A}(k)\mathbf{P}(k|k)\mathbf{A}'(k)+\mathbf{L}(k)\mathbf{Q}(k)\mathbf{L}'(k)$$

Kalman Filter Gain

$$\mathbf{K}(k+1)=\mathbf{P}(k+1|k)\mathbf{H}'(k+1)\left[\mathbf{H}(k+1)\mathbf{P}(k+1|k)\mathbf{H}'(k+1)+\mathbf{R}(k+1)\right]^{-1}$$

where

$$\mathbf{R}(k+1)=E\left[\mathbf{v}(k+1)\mathbf{v}'(k+1)\right]$$

Updated Covariance

$$\mathbf{P}(k+1|k+1)=\left[\mathbf{I}-\mathbf{K}(k+1)\mathbf{H}(k+1)\right]\mathbf{P}(k+1|k)$$

Updated State

$$\hat{\mathbf{x}}(k+1|k+1)=\hat{\mathbf{x}}(k+1|k)+\mathbf{K}(k+1)\Delta\mathbf{z}(k+1|k)$$
$$\Delta\mathbf{z}(k+1|k)=\mathbf{z}(k+1)-\mathbf{h}\left[\hat{\mathbf{x}}(k+1|k),k+1\right]$$

Figure 2. Extended Kalman Filter Equations

Another well known state of the art tracking technique is the interacting multiple model (IMM) filter [13, 14, 15, and 16]. The technique employs multiple models (a bank of Kalman filters) to perform state estimation. Each model may contain a different mathematical system model, observation model, variable dimensional state, or noise processes. For example, an IMM filter could consist of two Kalman filters, the first model using a straight line motion model with a low process noise and a second model with a high process noise. The two models are mixed probabilistically to fit the dynamics of the target. As long as the target follows a straight line motion and does not deviate from it, the first model will be in effect. As a target maneuvers from a straight line motion, the second model becomes in effect and in a sense the noisy positional measurements are connected together yielding a very poor velocity estimate of the target during the maneuver. The IMM architecture can also be used with an EKF used as one of its models. As before, to use the EKF we must know the nonlinearity of the system. When it comes to tracking moving targets we do not know what the craft will necessarily do next. This causes a problem with trying to model or account for a good motion model for the filter to use.

If a nonlinear model is unattainable, than a system identification technique could be used to create a model. A big buzz in the late 80's early 90's was artificial neural networks. A neural network is a function approximation technique. Given a set of inputs and a desired set of outputs a neural network could be trained to approximate well any function. The Stone Weierstrauss theorem states that given the class of squashing functions, we can uniformly approximate any continuous function [17]. This is purely an existence theorem. It does not state how to construct a neural network or how to train its weights. Neural networks contain a set of weights that must be determined to approximate functions. To train neural networks, techniques such as backpropagation [18, 19], evolutionary programming [20], and the extended Kalman filter [21] have been used. In [21], Singhal & Wu asked the question, "Why not use a Kalman filter to estimate the weights (states) of a neural network?" What they hypothesized was that the weights of a neural network are also the states of the network. The EKF training technique trains networks on the order of tens of iterations instead of thousands of iterations for such methods as backpropagation. An artificial network equation is shown in equation 1.

Equation 1.
$$NN_m = \sum_{i=1}^{N} w_{im} \cdot \left(f_i \left(\sum_{k=1}^{J} I_k * w_{ki} \right) \right)$$

where $f_i = \dfrac{1}{1 + \exp(-x_i)}$ is the output of the ith hidden node and x_i is the dot product sum of the previous input layer's output with the connecting weights of the hidden layer, NN_m is the mth output of the neural network, w_{im} is the mth output weight connected to the ith hidden node, w_{ki} is the kth input weight connected to the ith hidden node, and I_k is the kth input feeding the neural network. Stubberud took the EKF trainer one step further.

3. Neural Extended Kalman Filter

The Neural Extended Kalman Filter (NEKF) is based on the Singhal and Wu EKF neural network trainer. Stubberud took the technique one step further. Basically the algorithm is to use a Kalman filter to estimate the states by using a linear system model and at the same time use the Kalman filter to train a neural network to calculate the (nonlinearities, mismodeled dynamics, higher order modes, etc...) of the

system. Estimation of the system states are performed at once without the necessity of modeling the nonlinearities *a priori* as in the case of the Extended Kalman Filter. The NEKF architecture is shown in Figure 3. The inputs to the neural network are the updated states of the filter. The inputs are passed through an input layer, a hidden layer with nonlinear squashing functions, and an output layer. The outputs of the neural network are nonlinear corrections to the linear predicted state of the Kalman filter.

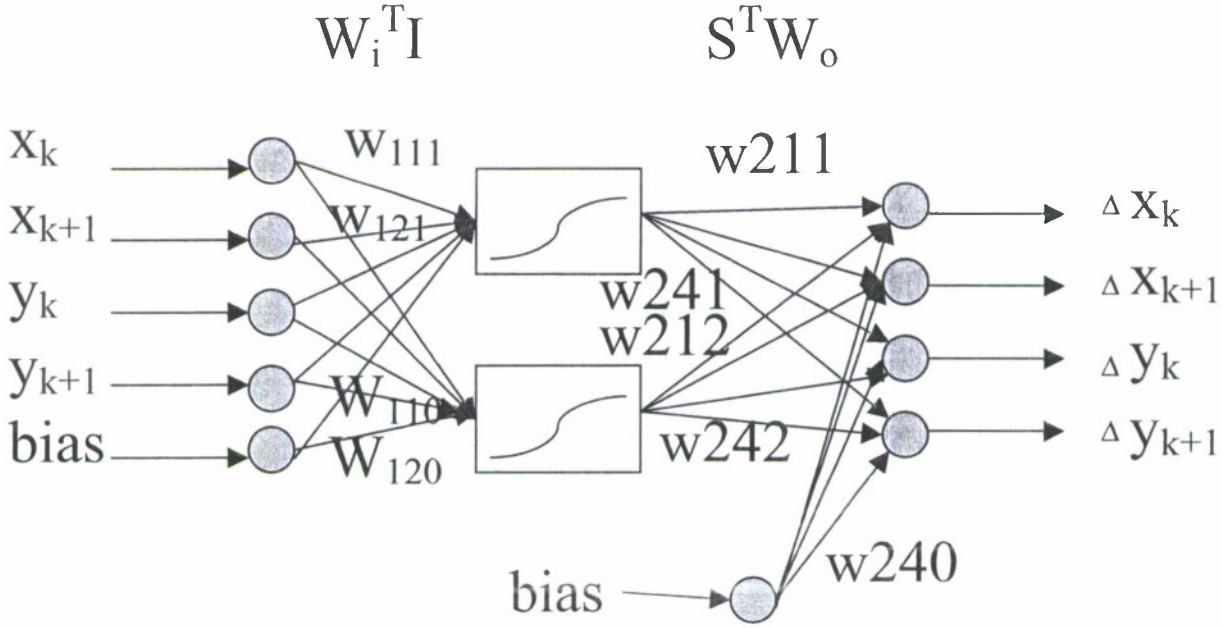

Figure 3. NEKF Architecture

The following are the NEKF Equations including the calculation of the Jacobian with respect to the neural network. This network is specifically for a 4 hidden node network with four states.

3.1. Predictor Equations for the NEKF

$$\hat{x}(k+1/k) = \hat{x}(k/k) + T_s \hat{v}_x(k/k) + NN1\left(\hat{\mathbf{x}}(k/k), \mathbf{w}(k/k)\right)$$

$$\hat{v}_x(k+1/k) = \hat{v}_x(k/k) + NN2\left(\hat{\mathbf{x}}(k/k), \mathbf{w}(k/k)\right)$$

$$\hat{y}(k+1/k) = \hat{y}(k/k) + T_s \hat{v}_y(k/k) + NN3\left(\hat{\mathbf{x}}(k/k), \mathbf{w}(k/k)\right)$$

$$\hat{v}_y(k+1/k) = \hat{v}_y(k/k) + NN4\left(\hat{\mathbf{x}}(k/k), \mathbf{w}(k/k)\right)$$

$$\mathbf{w}(k+1/k) = \mathbf{w}(k/k)$$

3.2. Neural Network Equations in NEKF

$$NN1\left(\hat{\mathbf{x}}(k/k),\mathbf{w}(\mathbf{k}/\mathbf{k})\right)=\sum_{j=1}^{4} w_{1j}^{2}\ f\left((\mathbf{w}_{j}^{1})^{T}\hat{\mathbf{x}}(k/k)+w_{j5}^{1}\right)+w_{15}^{2}$$

$$NN2\left(\hat{\mathbf{x}}(k/k),\mathbf{w}(\mathbf{k}/\mathbf{k})\right)=\sum_{j=1}^{4} w_{2j}^{2}\ f\left((\mathbf{w}_{j}^{1})^{T}\hat{\mathbf{x}}(k/k)+w_{j5}^{1}\right)+w_{25}^{2}$$

$$NN3\left(\hat{\mathbf{x}}(k/k),\mathbf{w}(\mathbf{k}/\mathbf{k})\right)=\sum_{j=1}^{4} w_{3j}^{2}\ f\left((\mathbf{w}_{j}^{1})^{T}\hat{\mathbf{x}}(k/k)+w_{j5}^{1}\right)+w_{35}^{2}$$

$$NN4\left(\hat{\mathbf{x}}(k/k),\mathbf{w}(\mathbf{k}/\mathbf{k})\right)=\sum_{j=1}^{4} w_{4j}^{2}\ f\left((\mathbf{w}_{j}^{1})^{T}\hat{\mathbf{x}}(k/k)+w_{j5}^{1}\right)+w_{45}^{2}$$

where

$$(\mathbf{w}_{j}^{1})^{T}=\begin{bmatrix} w_{j1}^{1} & w_{j2}^{1} & w_{j3}^{1} & w_{j4}^{1}\end{bmatrix};\quad j=1,2,3,4$$

$$\hat{\mathbf{x}}(k/k)=\begin{bmatrix}\hat{x}(k/k) & \hat{v}_{x}(k/k) & \hat{y}(k/k) & \hat{v}_{y}(k/k)\end{bmatrix}$$

3.3. The NEKF State Vector

Define

$$(\mathbf{w}_{j}^{2})^{T}=\begin{bmatrix} w_{j1}^{2} & w_{j2}^{2} & w_{j3}^{2} & w_{j4}^{2}\end{bmatrix};\quad j=1,2,3,4$$

$$(\mathbf{w}_{5}^{1})^{T}=\begin{bmatrix} w_{15}^{1} & w_{25}^{1} & w_{35}^{1} & w_{45}^{1}\end{bmatrix}$$

$$(\mathbf{w}_{5}^{2})^{T}=\begin{bmatrix} w_{15}^{2} & w_{25}^{2} & w_{35}^{2} & w_{45}^{2}\end{bmatrix}$$

Note that the notation for $(\mathbf{w}_{5}^{1})^{T}$ and $(\mathbf{w}_{5}^{2})^{T}$ is different from that for $(\mathbf{w}_{j}^{1})^{T}$ and $(\mathbf{w}_{j}^{2})^{T}$ for $j=1,2,3,4$.

The NEKF state vector can be written as

$$\bar{\mathbf{x}}(k/k) = \begin{bmatrix} \hat{\mathbf{x}}(k/k) \\ \mathbf{w}(k/k) \end{bmatrix}$$

where

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1^1 \\ \mathbf{w}_2^1 \\ \mathbf{w}_3^1 \\ \mathbf{w}_4^1 \\ \mathbf{w}_5^1 \\ \mathbf{w}_1^2 \\ \mathbf{w}_2^2 \\ \mathbf{w}_3^2 \\ \mathbf{w}_4^2 \\ \mathbf{w}_5^2 \end{bmatrix}$$

Note that $\mathbf{w}_1^1$, $\mathbf{w}_2^1$, $\mathbf{w}_3^1$, and $\mathbf{w}_4^1$ represent 16 weights in the hidden layer of the neural network, $\mathbf{w}_5^1$ represents 4 biases in the hidden layer of the neural network, $\mathbf{w}_1^2$, $\mathbf{w}_2^2$, $\mathbf{w}_3^2$, and $\mathbf{w}_4^2$ represent 16 weights in the output layer of the neural network, and $\mathbf{w}_5^2$ represents 4 biases in the output layer of the neural network.

Thus the vector $\mathbf{w}$ has 40 elements and the state vector of the NEKF has 44 elements.

Note that the prediction vector is given by

$$\bar{\mathbf{x}}(k+1/k) = \begin{bmatrix} \hat{\mathbf{x}}(k+1/k) \\ \mathbf{w}(k+1/k) \end{bmatrix}$$

3.4. The Jacobian Matrix

The Jacobian matrix $\mathbf{J}$ is a 44×44 matrix generated by differentiating the vector $\bar{\mathbf{x}}(k+1/k)$ by the vector $\bar{\mathbf{x}}(k/k)$, that is,

$$\mathbf{J} = \frac{\partial \mathbf{x}(k+1/k)}{\partial \mathbf{x}(k/k)}$$

We now make the following definitions

$$\mathbf{W}^1 = \begin{bmatrix} (\mathbf{w}_1^1)^T \\ (\mathbf{w}_2^1)^T \\ (\mathbf{w}_3^1)^T \\ (\mathbf{w}_4^1)^T \end{bmatrix} \qquad \mathbf{W}^2 = \begin{bmatrix} (\mathbf{w}_1^2)^T \\ (\mathbf{w}_2^2)^T \\ (\mathbf{w}_3^2)^T \\ (\mathbf{w}_4^2)^T \end{bmatrix}$$

$$f'(z_j) = \left.\frac{\partial f(z)}{\partial z}\right|_{z=z_j} \qquad \text{where } z_j = (\mathbf{w}_j^1)^T \hat{\mathbf{x}}(k/k) + w_{j5}^1$$

$$\mathbf{F}'(\mathbf{z}) = \begin{bmatrix} f'(z_1) & 0 & 0 & 0 \\ 0 & f'(z_2) & 0 & 0 \\ 0 & 0 & f'(z_3) & 0 \\ 0 & 0 & 0 & f'(z_4) \end{bmatrix}; \qquad \mathbf{z} = [z_1 \;\; z_2 \;\; z_3 \;\; z_4]$$

$$f(z_j) = f(z)\Big|_{z=z_j}$$

$$\mathbf{f}^T(\mathbf{z}) = \begin{bmatrix} f(z_1) & f(z_2) & f(z_3) & f(z_4) \end{bmatrix}$$

$$\mathbf{F}(\mathbf{z}) = \begin{bmatrix} \mathbf{f}^T(\mathbf{z}) & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{f}^T(\mathbf{z}) & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{f}^T(\mathbf{z}) & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{f}^T(\mathbf{z}) \end{bmatrix}$$

where $\mathbf{0}^T = [0 \;\; 0 \;\; 0 \;\; 0]$

$$\mathbf{X}(k/k) = \begin{bmatrix} \hat{\mathbf{x}}^T(k/k) & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & \hat{\mathbf{x}}^T(k/k) & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & \hat{\mathbf{x}}^T(k/k) & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \hat{\mathbf{x}}^T(k/k) \end{bmatrix}$$

$\mathbf{O}_{n\times m}$ = zero matrix with n rows and m columns

$\mathbf{I}_{n\times n}$ = identity matrix with n rows and columns

Using these definitions, the Jaeobian matrix ean be written as

$$\mathbf{J} = \left[\begin{array}{c|c} \mathbf{A} + \mathbf{W}^2\mathbf{F}'(\mathbf{z})\mathbf{W}^1 & \mathbf{W}^2\mathbf{F}'(\mathbf{z})\mathbf{X}(k/k) \mid \mathbf{W}^2\mathbf{F}'(\mathbf{z}) \mid \mathbf{F}(\mathbf{z}) \vdots \mathbf{I}_{4\times 4} \\ \hline \mathbf{O}_{40\times 4} & \mathbf{I}_{40\times 40} \end{array}\right]$$

4. Tracking result comparisons

The following is a set of comparison results between a linear Kalman filter, a linear IMM 2 model filter, and two NEKF's with 4 and 10 hidden nodes, respeetively. To make the eomparisons between the tracking filters eaeh filter used the same amount of proeess noise, except for the IMM where its first model used none. Each filter used a eonstant velocity straight line motion model with a 4 state veetor eonsisting of position and veloeity in a Cartesian eoordinate system. The neural network arehiteeture for both networks only used 2 outputs in this simulation. These two outputs were nonlinear or unmodeled higher order mode eorreetions to the veloeity states. The nonlinear squashing funetions used by the neural networks here hyperbolie tangent functions. The network with 4 hidden nodes had a total of 30 weights to be trained at eaeh time sample for adaptation and the network with 10 hidden nodes had 72 weights. This tracking study was for a single target on a two dimensional Cartesian grid. A jerk motion model was used to generate truth. This was used to yield a smoother ramp up for aeeeleration and veloeity for a more realistie scenario. Figure 4 shows the true position of the track for the seenario. The track starts at (0,0) in the grid at a speed of 10 units/time sample along the x-axis. It then inereases its speed to 35 units/time sample and then deeelerates back to 10. Then the traek turns right and aeeelerates through a 180 degree turn. Its top speed though the turn is 90 units/time sample. The traek then deeelerates baek to 26 units/time sample. A left 180 degree turn is then performed moving the traek baek on the positive x-axis direetion. The track then deeelerates again to 20 units/time sample. Finally, the traek moves into five 360 degree turns before the seenario ends. A radar equation was implemented for the simulation to generate range and bearing measurements from the sensor to the target. Gaussian noise was added to both the true range and true bearing to be used by eaeh filter. The amount of uncertainty used varied from 10, 50, and 100 units of noise for range and 1, 2, 3, and 10 degrees for bearing. The sensor location in this scenario is loeated at the origin. The sensor loeation eaused the beginning data of the target to be very noisy due to the measurements all falling very elose into the sensor. As the target's trajeetory moved away from the origin the data began to look more like the trajeetory and not just bouneing around aimlessly. Figure 5 is shown to provide an idea of noisy range and bearing measurements. They have been converted to x,y eoordinates for display purposes. In the figure a Monte-Carlo Kalman filter result is overlayed in blue. Notice the large lag when the aeeelerating turn oecurs and

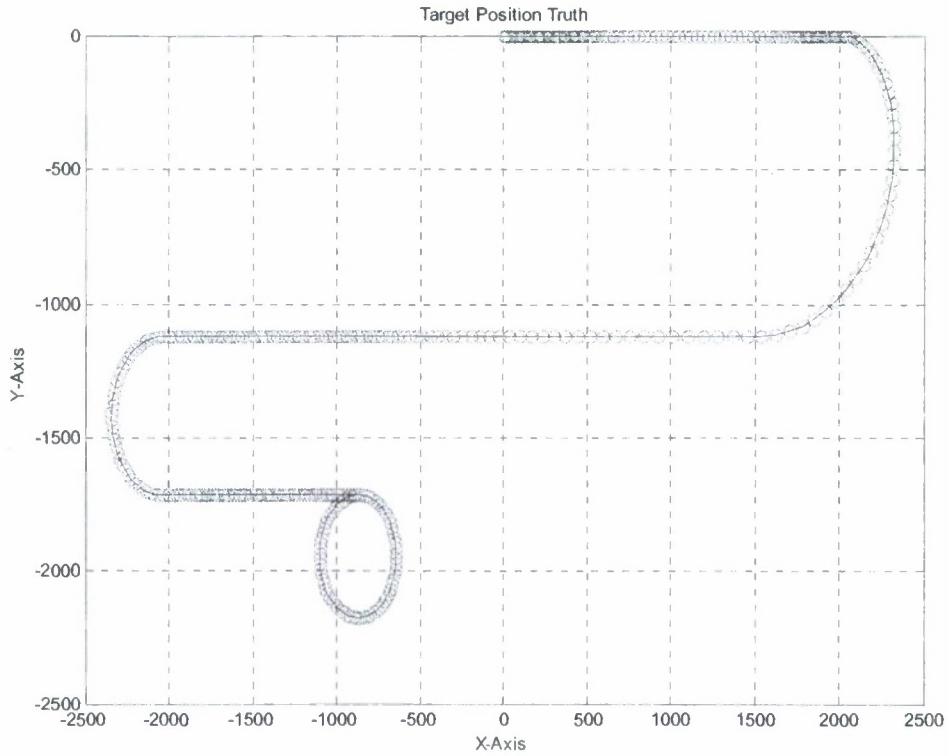

Figure 4. Scenario

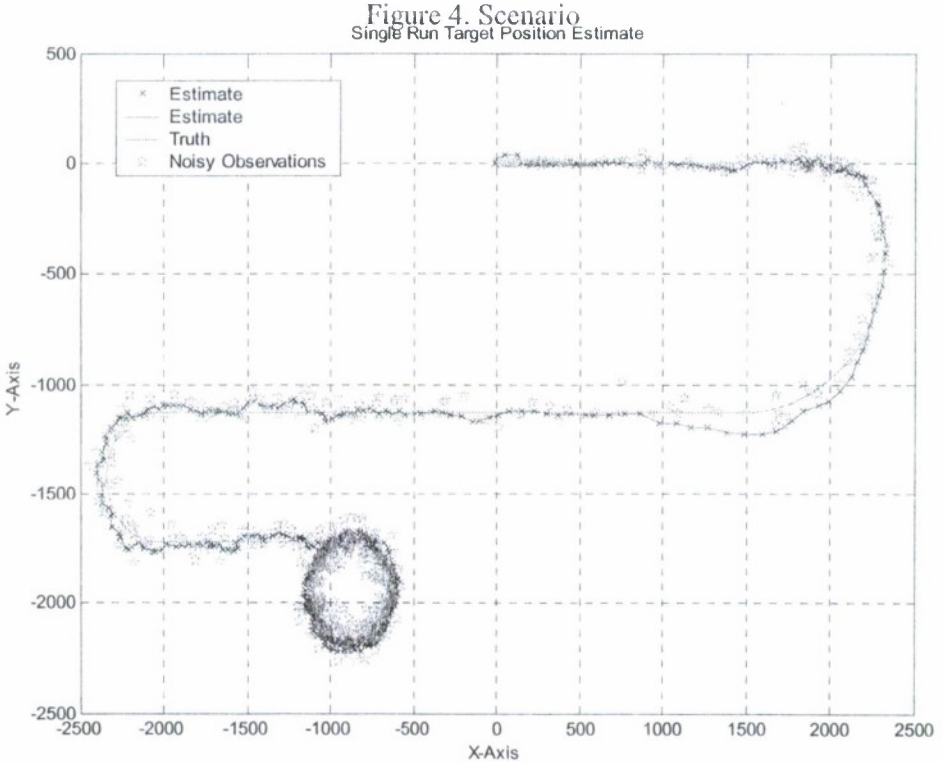

Figure 5. Noisy Observations

when the constant velocity turns occur. Figure 6 is a zoomed in view of the 5 circular turns of the target that look like a hornet's nest. This figure shows another result from the Kalman filter. The range/bearing error on the measurements can be seen in this figure. Notice the right and left sides of the track lag heavily and the top and bottom track rather well close to truth. This is due to the target trajectory in reference to the sensor location. Figures 7-9 are results from a typical network run. Figure 7 shows the weights of a four hidden node neural network. The number of weights in the neural network are 30 for this network. The large swings in network adaptation are due to the target maneuvers causing the network to add nonlinear corrections to the output of the linear Kalman filter. Of note, the neural network is also adding unmodeled mode dynamics such as acceleration corrections or jerk corrections to adjust the velocity states. For example, if in the real target track an acceleration is increasing the velocity in the x direction by two, the neural network will adapt and increase the velocity estimate by two units. The neural network adapts through the residual error between the predicted observations and the noisy observations from the sensor. In Figure 7 the first real large

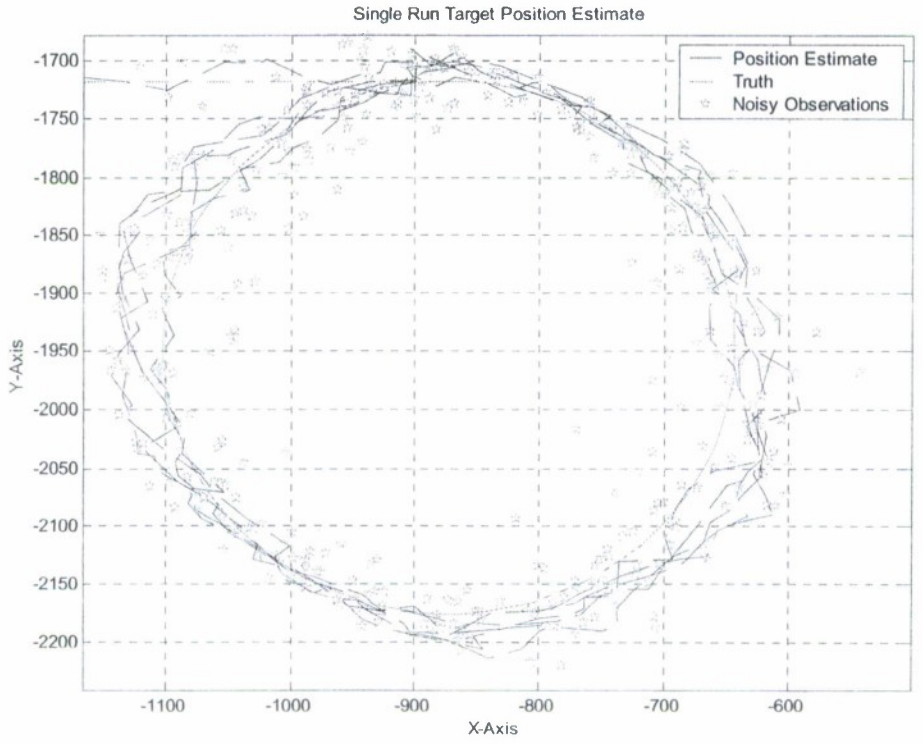

Figure 6. Hornet's Nest

Adaptation occurs at time 50. This is when the first acceleration is performed by the target. At time 80 the target decelerates back to its original speed and the network adapts again. At time 120 a very large adaptation occurs in the neural network. At this time, the target is accelerating and performing a 180 degree turn. The speed increases by 9 times by the end of this maneuver from 10 units/time sample to 90 units/time sample. The network has to keep adjusting as each data point is processed to keep up with the large maneuver. At time 170 the target then decelerates back to a speed of 26 units/time sample. At time 230 the second 180 degree turn occurs and the network adapts again. At time 275 another deceleration

occurs and the network adapts again. Finally, at time 340 until the end of the scenario the target's trajectory is a constant speed 5 degree/time sample turn. From this time on in the figure it can be seen that the neural network weights are adapting in a sinusoidal fashion to accommodate the corrections to the x and y velocity states. Another note about Figure 7 is the weights that almost look constant throughout the run. These weights are the bias weights in the network's input and output layers. These weights perform noticeable adjustments when strong maneuvers occur in the trajectory. These bias weights are needed to perform an affine transformation to move the function approximation from having to exist always at the origin. A final note about this figure is since the observations are noisy the weights are always constantly adapting. The sinusoidal inputs look noisy because of this. The better the sensor the

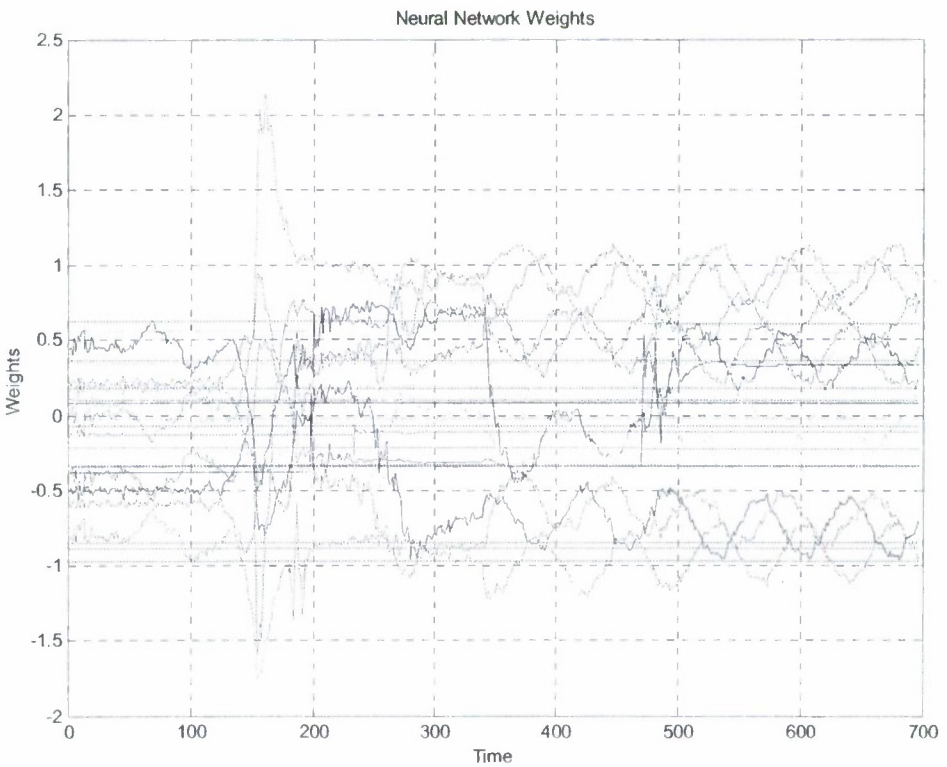

Figure 7. NEKF Weights

less the weights have chatter in their adaptation. Figures 8 and 9 are the outputs for the y and x velocity corrections for the neural network, respectively. Recalling that the first maneuvers at time 50 and 85 only occur in the x-direction, Figure 8 shows that the corrections for the y-velocity state are very small and in the noise. Figure 9, however, contains adjustments for the x-velocity where the maneuvers occur. At these times the network increases the x-velocity and then decreases when the target slows down again. When the accelerating turn occurs at time 120 the network's outputs for x and y velocity corrections are the largest. When the maneuvers stop and the trajectory returns to straight line motion the network's outputs go to small levels. Notice at the end of the figures the sinusoidal corrections for adapting through the circular motion.

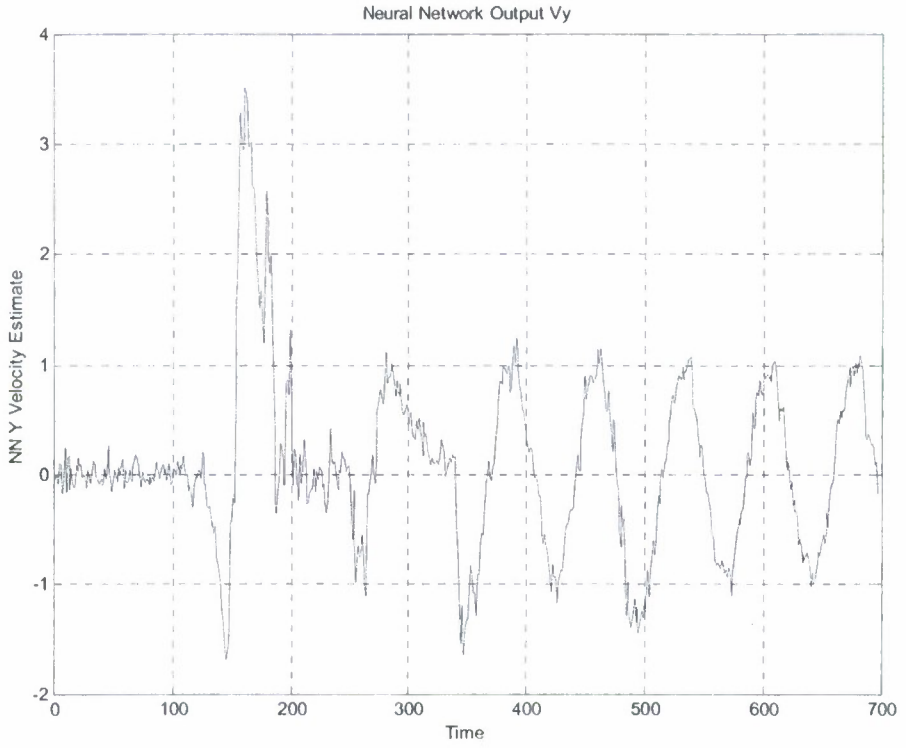

Figure 8. Y Velocity Correction

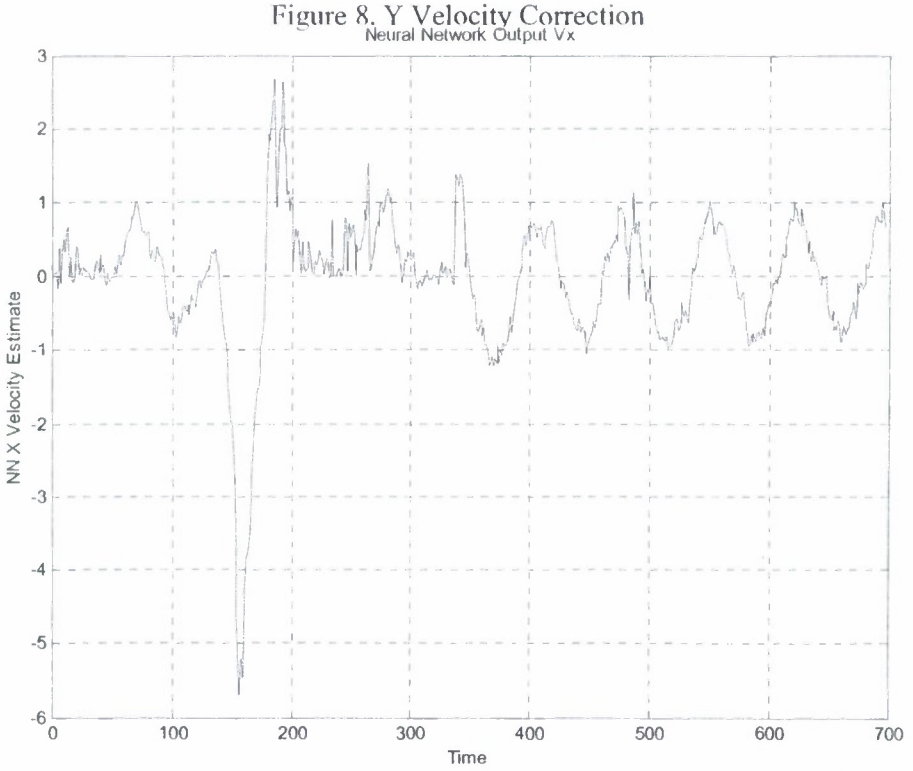

Figure 9. X Velocity Correction

For each of the tracking results in Figures 10-15 a time average position error statistic of 50 Monte-Carlo runs was used to generate the plots. The statistic was calculated by the following four equations

$$\mathbf{pos\_err} = \mathbf{pos\_err} + \sqrt{(\hat{\mathbf{x}} - xtruth)^2 + (\hat{\mathbf{y}} - ytruth)^2}$$

$$\mathbf{noise\_err} = \mathbf{noise\_err} + \sqrt{(z_x - xtruth)^2 + (z_y - ytruth)^2}$$

$$\mathbf{Delta} = \mathbf{noise\_err} - \mathbf{pos\_error}$$

$$avg(i) = avg(i+1) + Delta(i)$$

where **pos_err** is the position error between the state estimates and truth, **noise_err** is the position error between the noisy observations and truth, **Delta** is the difference between the noise error and the estimate error, and **avg** is the time average of the error statistic. If the **avg** is increasing the filter is doing well, and if the **avg** is decreasing the filter is performing poorly. Fifty Monte-Carlo runs were averaged for each error statistic that is plotted in Figures 10-15. In each figure all four tracking filters are plotted

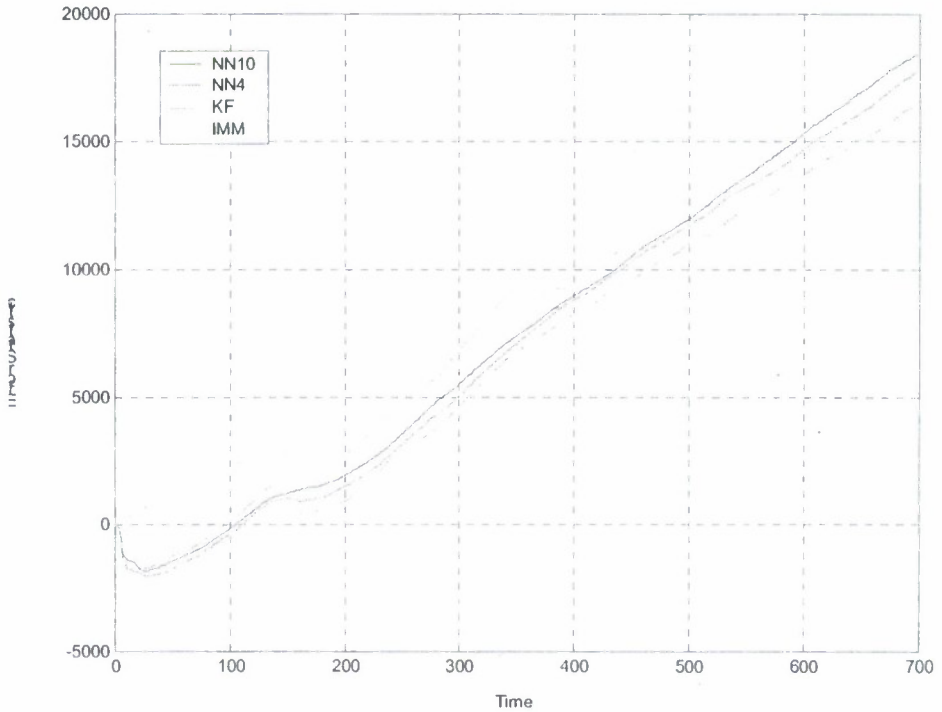

Figure 10. R = 50, B = 2

together and overlayed on the same plot. Each plot has a legend where the network with 10 hidden nodes is blue, the network with 4 hidden nodes is red, the Kalman filter is magenta, and the IMM is green. Figure 10 is a result where the range uncertainty was set to 50 units, and the bearing uncertainty to 2 degrees. In this figure the IMM is ahead of the NEKF results until the hornet's nest maneuver. Then both NEKF's perform much better than both the Kalman filter and the IMM. In Figure 11, the bearing

uncertainty is now set to 1 degree. The NEKF's once again outperform the IMM and the Kalman filter. In Figure 12 the range uncertainty was increased to 100 units. In this case the NEKF's performed the worst compared to the Kalman filter and the IMM. This was due to the initial measurements from the sensor having large amounts of noise and not allowing a trajectory to form until the target moved far enough away from the origin. It is believed that if the scenario would have run longer that the NEKF's would have caught up with the other estimators. Notice at the end of the error statistic plots on Figure 12 that the NEKF results are increasing and the IMM and Kalman filter are starting to decrease. A box car average has been suggested to better compare the different estimators.

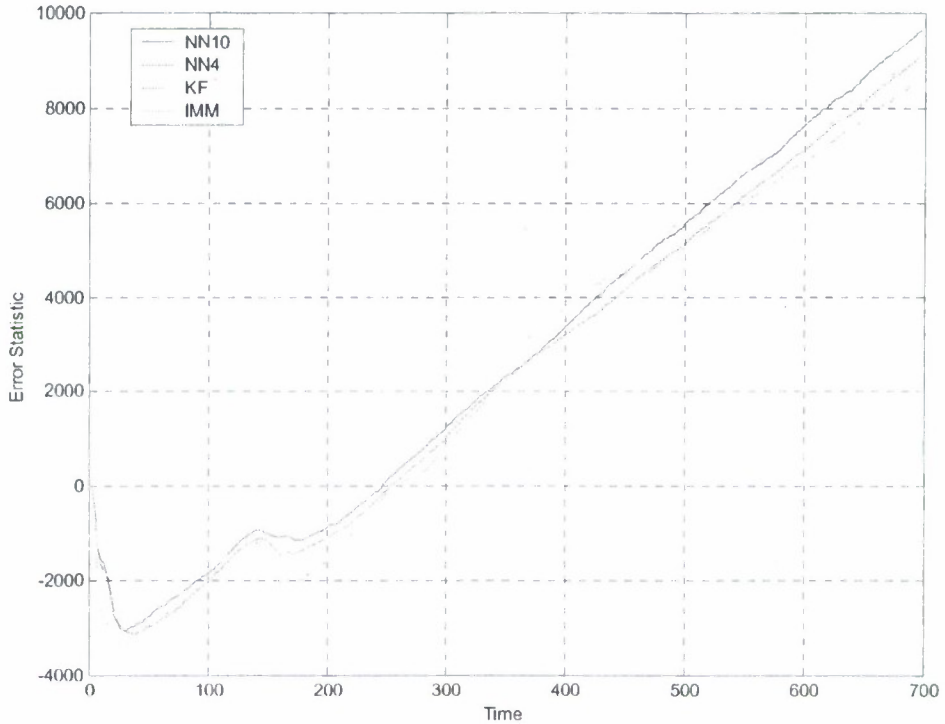

Figure 11. R = 50, B = 1

Figure 13 shows the results of setting the range uncertainty to 10 units along with the bearing uncertainty still set to 1 degree. This plot more clearly shows the NEKF's outperforming the other two filters. Figure 14 shows the results of setting the range uncertainty to 100 units with the bearing uncertainty set to 3 degrees. Once again in this figure the NEKF's outperform the other two trackers. Finally, in Figure 15 the results of setting the range uncertainty to 100 units with the bearing uncertainty set to 10 degrees is shown. This figure shows the NEKF's lower in score, but all of the filter results are very close together in this highly noisy case.

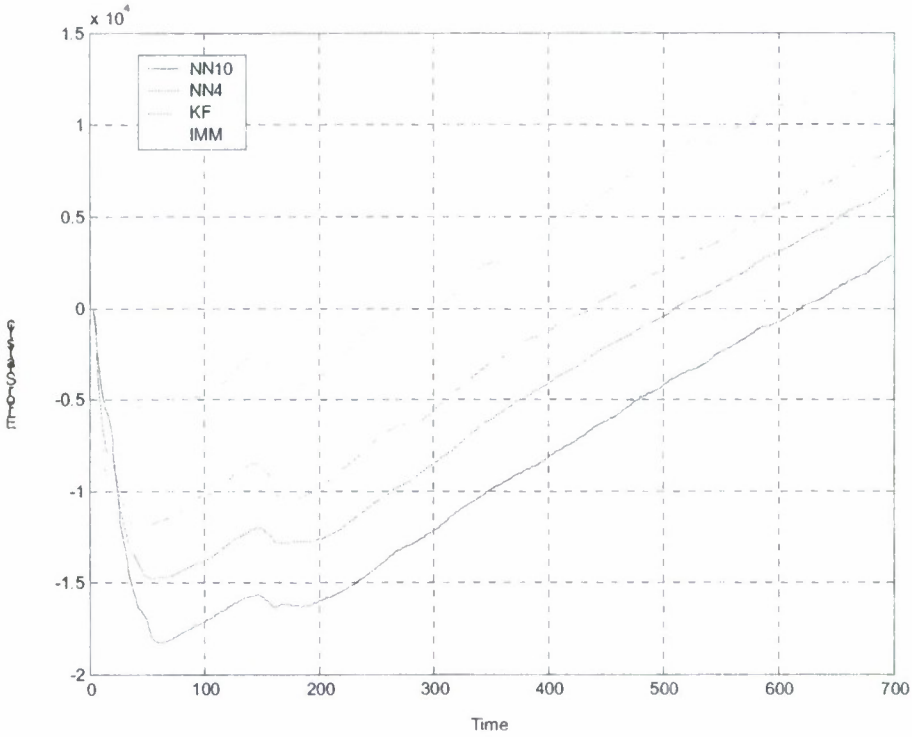

Figure 12. R = 100, B = 1

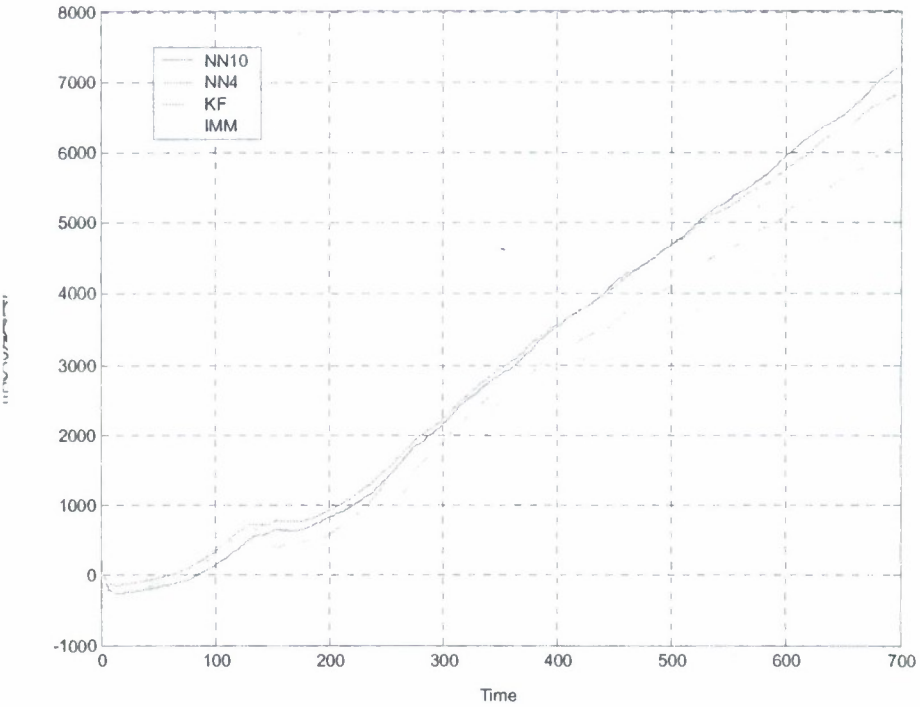

Figure 13. R = 10, B = 1

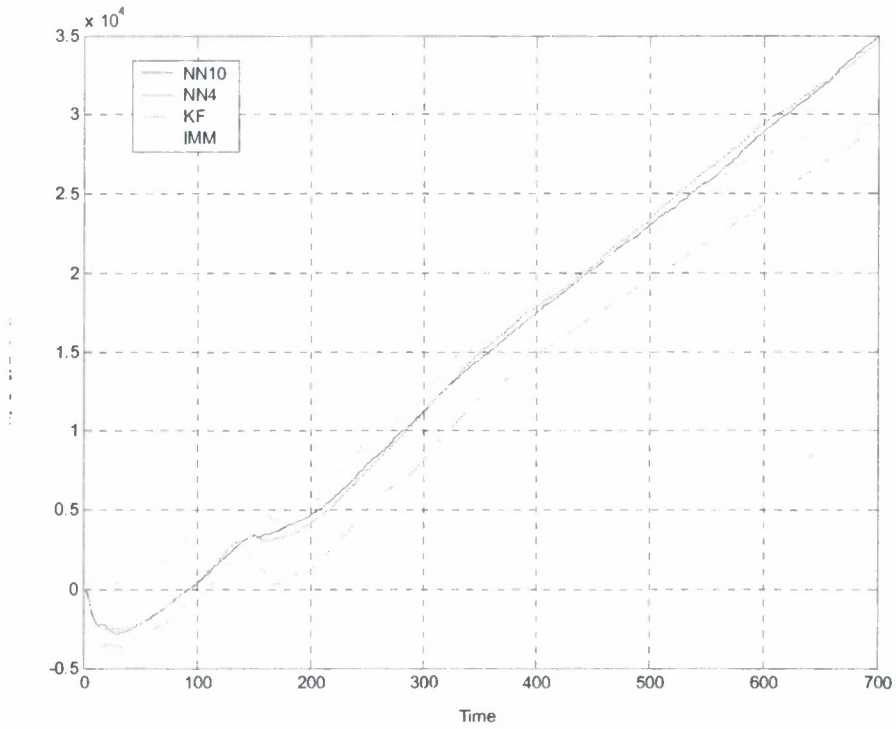

Figure 14. R = 100, B = 3

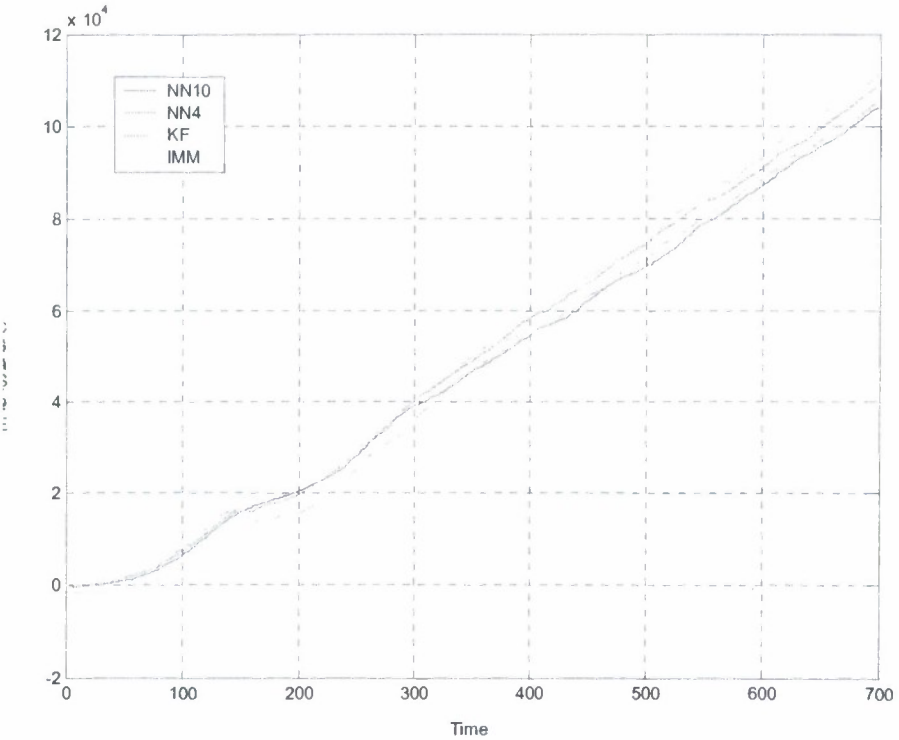

Figure 15. R = 100, B = 10

5. Research Issues

Some of the research issues this project is trying to address is as follows.

- Show that the Neural Extended Kalman Filter can be embedded into an Interacting Multiple Model (NEKF IMM) algorithm
- Show that the NEKF IMM algorithm provides smoothed estimates during system dynamics
- Show that the NEKF provides smoothed estimates during system dynamics
- Show that the NEKF has a second order convergence
- Develop techniques for choice of optimal noise processes of the NEKF
- Determine the neural network architecture needed to work across a broad range of problems
- Show improved tracking performance over other techniques (alpha-beta, Kalman, IEKF, IMM, etc…)
- Investigate the effects of the NEKF in the IMM architecture
- Targeting specific states (velocity) or is all more robust
- Observability dependence on initial conditions
- Observability can lead to a Stability Proof
- Multi-Target Tracking
- Data Association
- Pd, Pfa, Loss of Signal Detection making a more complex realistic scenario
- Squashing Functions
- Multi-Layer Networks
- Is an IMM architecture required
- Tracking Benchmark Study
- Higher Order Models
- Data Fusion in a Multi-Sensor Environment

6. Conclusions

In this paper we discussed the neural extended Kalman filter as applied to the target tracking problem. The NEKF uses a neural network to adapt on-line to unmodeled dynamics or nonlinearities in the target trajectory. This on-line adaptation provides for a robust state estimation for tracking applications because the maneuvers do not have to be known beforehand. The NEKF is a generic state estimator that can be

used to estimate any state vector such as position, velocity, magnetic moment, frequency signatures, etc... Comparisons were performed between a linear Kalman filter, a 2 model IMM filter, and two NEKF's with differing hidden layers. The NEKF outperformed the other filters in almost every case analyzed. Further investigation into why the NEKF did not outperform in each case is now under investigation.

7. Acknowledgements

The authors would like to thank the SPAWAR Systems Center San Diego In-House Laboratory Independent Research (ILIR) program funded through the Office of Naval Research for funding this effort.

8. References

[1] *Robust Tracking with a Neural Extended Kalman Filter* – Office of Naval Research (ONR) In-House Laboratory Independent Research (ILIR) Project FY02

[2] "Approximation and Estimation Techniques for Neural Networks," A. R. Stubberud, H. Wabgaonkar, *Proceedings of the 28th Conference on Decision and Control*, Honolulu, Hawaii, pp 2736-2740, December, 1990.

[3] "A Neural-Network-Based System Identifcation Technique," A. R. Stubberud, H. Wabgaonkar and S.C. Stubberud, *Proceedings of the 30th Conference on Decision and Control*, Brighton, England, pp 869-870, December, 1991.

[4] "Adaptive Extended Kalman Filter Using Artificial Neural Networks," R.N. Lobbia, S.C. Stubberud, and M.W. Owen, *The International Journal of Smart Engineering System Design,* Vol. 1, pp. 207-221, 1998.

[5] "Artificial Neural Network Feedback Loop," S.C. Stubberud, M.W. Owen, *Proceedings of the 11th International Symposium on Intelligent Control,* Dearborn, Michigan, pp. 514-519, September, 1996.

[6] "An Adaptive Extended Kalman Filter Using Artificial Neural Networks," R. N Lobbia, S.C. Stubberud, and M.W. Owen, *Proceedings of the 34th Conference on Decision and Control*, New Orleans, Louisiana, pp 1842-1846, December, 1995.

[7] M. Owen, "Application of Neural Networks Using Extended Kalman Filtering", Master Thesis, California State University Long Beach, May 1997.

[8] S. Haykin, Neural Networks and Kalman Filtering, Wiley & Sons, 2001.

[9] S. Blackman, Multiple-Target Tracking with Radar Applications, Artech House, 1986.

[10] S. Blackman and R. Popoli, Design and Analysis of Modern Tracking Systems, Artech House, 1999.

[11] A. Gelb, Applied Optimal Estimation, M.I.T. Press, 1974.

[12] M. Santina, A. Stubberud, and G. Hostetter, Digital Control System Design, Saunders College Publishing, 1994.

[13] Y. Bar-Shalom and X. Li, Estimation and Tracking: Principles, Techniques, and Software, Artech House, 1993.

[14] Y. Bar-Shalom and X. Li, Multitarget-Multisensor Tracking: Principles and Techniques, YSB Publishing, 1995.

[15] Y. Bar-Shalom and X. Li, and T. Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, 2001.

[16] Y. Bar-Shalom and W. Blair, Multitarget-Multisensor Tracking: Applications and Advances Volume III, Artech House, 2000.

[17] A. R. Stubberud, H. Wabgaonkar, "A Neural Network Based Non-linear Control Technique", Advances in Control, C. T. Leondes (ed.) Academic Press, 1992.

[18] (eds.) D. White and D. Sofge, Handbook of Intelligent Control Neural, Fuzzy, and Adaptive Approaches, Van Nostrand Reinhold, 1992.

[19] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley Publishing Company, 1990.

[20] D. B. Fogel, Evolutionary Computation Toward a New Philosophy of Machine Intelligence, IEEE Press, 1995.

[21] S. Singhal and L. Wu, "Training Multilayer Perceptrons with the Extended Kalman Algorithm," Advances in Neural Information Processing System I, D.S. Touretzky (ed.) Morgan Kaufmann, 1989, pages 133-140.